The Microsoft Hypervisor

Marcus Peinado

Architect
Virtual Machine Technologies Group

Changes since last August

- Recall: NGSCB runs two isolated operating systems on a single machine.
- We are building a general-purpose hypervisor.
  - For NGSCB and for other applications
  - Work started in April
- TCAAB recommendations from last year
  - Build a general purpose VMM
  - Do not multiplex I/O resource in it.
- Other changes to NGSCB
  - Will be discussed in the next talk.

General Remarks

- **The hypervisor is work in progress**
  - We have been working on it for about three months.
  - Many details are still TBD.
  - Ideal time for feedback and advice.
  - I will be asking for advice in specific areas.

The Microsoft Hypervisor

- Motivation and Goals
- Architectural Overview
- Development Process and Assurance
- Detailed design
  - CPU
  - Memory
  - Control Transfer
  - Interrupts
  - IPC
  - Access Control

Motivation and Goals

Terminology

- Hypervisor
  - Layer of software that sits just above the hardware and beneath one or more operating systems
  - Its primary job is to provide isolated execution environments for software.
- Partition
  - An isolated execution environment provided by the hypervisor
- Guests
  - Any software running within a partition

Example

- Typical guests: operating systems
  - May be "unmodified off-the-shelf"
- Typical configuration: several operating systems execute independently on the same machine
  - In hypervisor partitions
- Partition: Looks to a guest OS like a real machine
  - CPU, memory, interrupts (hypervisor)
  - Concrete devices, such as disk (special guests)

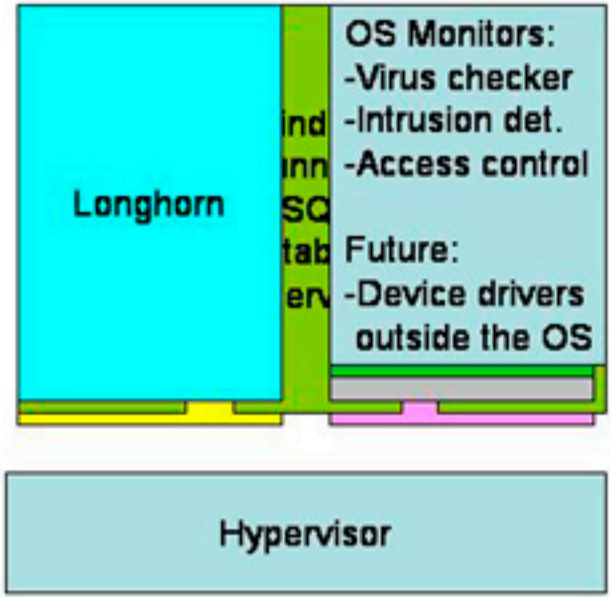
Some related work

- 1970s VMMs for mainframes [IBM]
- Some 'orange book' systems
  - A1 VMM at DEC [SVS]
  - SCOMP
- Different kinds of small isolation kernels
  - Exokernel
  - Denali
- VMMs for personal computers
  - XEN
  - Virtual PC / Connectix
  - VMware

The problem we are trying to solve

Target use areas

- Security / NGSCB
- Server consolidation
- Legacy support
- Consumer electronics applications
- Operating system robustness



Desired Properties

- Isolation
  - More on the next slide
- Performance
  - Some applications are performance critical (e.g. server consolidation)
- Assurance
  - Hard to quantify
  - Significantly better than the assurance in existing Microsoft products.
  - Management guidance against methodologies that
    - Add significant cost to the development process
    - Are incompatible with the Microsoft environment

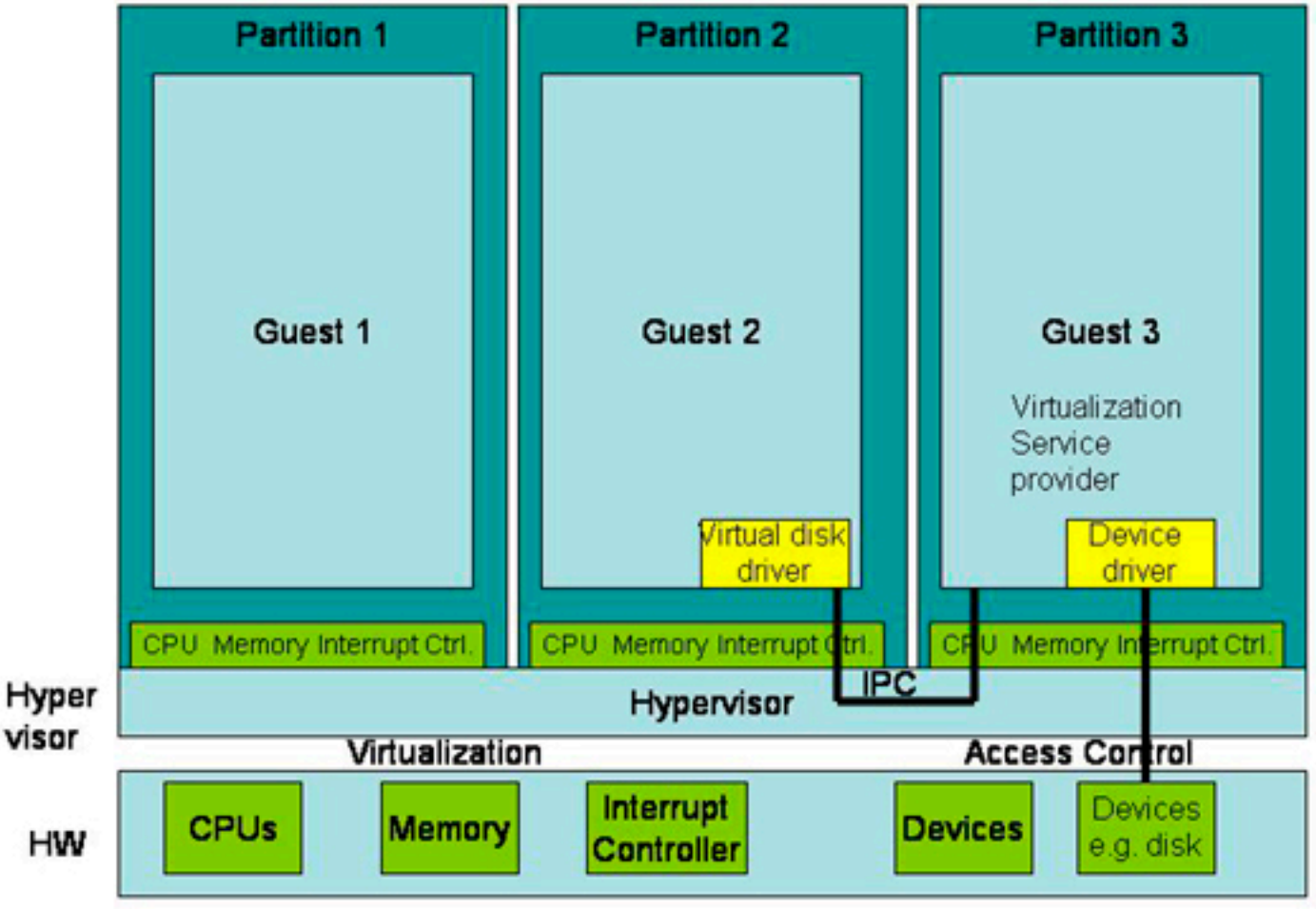
Isolation Goals

- Confidentiality
  - No collection of guests should be able to observe the computations of another guest (unless authorized).
- Integrity
  - No collection of guests should be able to interfere with the computations of another guest (unless authorized).
- Availability
  - No collection of guests should be able to prevent another guest from executing (unless authorized).
  - The first version of the hypervisor will not meet this goal.

Architectural Overview

Extensibility

- Hypervisor executes at the highest CPU privilege level.
- Extensibility
  - The hypervisor is monolithic: no plug-ins, loadable kernel extensions, device drivers in the hypervisor
  - Any extension executes as a guest in a regular partition
    - Hypervisor provides IPC mechanisms
    - cf. microkernels
    - New hardware constrains DMA devices
- Examples of external components
  - Device virtualization (virtualization service providers)
  - Administration and policy configuration


Partition 1
Partition 2
Partition 3
Guest 1
Guest 2
Guest 3
Virtualization
Service
provider
Virtual disk
driver
Device
driver
CPU Memory Interrupt Ctrl.
CPU Memory Interrupt Ctrl.
CPU Memory Interrupt Ctrl.
IPC
Hyper
visor
Hypervisor
Virtualization
Access Control
HW
CPUs
Memory
Interrupt
Controller
Devices
Devices
e.g. disk

Partitions and Virtual Processors

- Address space
  - Each partition exposes a contiguous zero-based physical address space
  - This is not the real physical address space of the machine
- Virtual Processors
  - Each partition exposes one or more virtual processors (VP)
    - Each VP looks like a real x86 to the guest
    - Scheduled on physical processors
- Virtual interrupt controller
- Guest types
  - Unmodified off-the-shelf
  - "enlightened" guests that are aware of the hypervisor ("para-virtualization")
    - Device drivers
    - Kernel modifications

Control transfers

- Explicit guest call into the hypervisor
  - The hypervisor exposes a programmatic interface
    - Configuration
    - Performance optimization
- Guest trap into the hypervisor (intercept)
  - Possible causes:
    - Guest executed privileged instruction
    - Page fault etc.
- Interrupt
  - The hypervisor receives all interrupts
  - The hypervisor exposes a virtual interrupt controller in each partition.

Device Management

- Goal
  - Enable arbitrary assignment of devices to partitions
  - Protection from DMA devices
  - Real physical addresses opaque to guests
- Problem
  - Existing DMA hardware protections are insufficient.
  - DMA devices operate on physical addresses.
- Hardware vision
  - Second-level address translation
  - Make DMA devices operate on "virtual" addresses
  - Use standard techniques to constrain them transparently.

Limitations of the Version 1 Product

- The V1 Hypervisor will heavily rely on a special partition (primary partition) running Longhorn
- Scheduling
  - The V1 hypervisor will not have its own scheduler
  - It will use the scheduler of the primary partition
- Memory management
  - The hypervisor has to acquire memory from the primary partition
- Device management
  - The hypervisor assigns (almost) all devices to the primary partition.
  - The primary partition runs in V=R mode.
- Goal: The limitations should be "transparent" to guests.
- Reasons for the limitations
  - Our product schedule
  - Hardware availability

Assurance Measures

- Precise specifications
  - Focus on externally visible behavior
    - of the hypervisor
    - of subcomponents of the hypervisor
  - Currently very limited formal methods
- Layered design [SVS90]
  - Still working on the layering
- Keep the system small
  - TCB minimization
- Array of other measures
  - Penetration testing, TMA, coding standards, code reviews, ...
  - Details in the HASE talk

Design

- CPU
- Memory
- Interrupts
- IPC
- Access Control
- Control Transfer

CPU

- Virtual processor (VP) abstraction exposed to guests
- Each VP behaves like an x86 CPU
  - Hypervisor has to emulate certain privileged instructions and CPU state
  - Hypervisor enables external handler for certain instructions (e.g. CPUID), traps etc.
  - Interfaces to read and write VP state
- VP states
  - Ready
  - Running
  - Halted
  - Blocked

Memory

- Each partition exposes
  - a zero-based contiguous physical address space.
  - an x86-compatible MMU abstraction
  - an optimized MMU abstraction with tagged TLBs
- Machine physical addresses are hidden from guests (except primary V=R guest)
- Two levels of address translation
  - Guest virtual → guest physical
    - Given by x86-compatible MMU abstraction
  - Guest physical → machine physical
    - Produces zero-based contiguous guest address spaces
    - Implemented by shadow page table algorithm or
    - SLAT hardware if/when available

Interrupts

- Hypervisor takes all interrupts
  - Programs hardware interrupt controller (APIC)
  - Exposes a virtual interrupt controller (synthetic interrupt controller SynIC) to each partition
- SynIC functions
  - Local APIC compatible interface for legacy guests
  - Modified interface for improved performance
  - Inter partition "signals"
    - May have small attached messages

IPC

- SynIC "signals"
- Shared memory
  - Guests can build arbitrary high-performance IPC mechanisms over this primitive.
- Messages
  - Fixed bound on message size
  - Primary purpose: help set up shared memory
  - Blocking and non-blocking send
    - Message queuing in the hypervisor
  - Asynchronous receive

Access Control

- Hypervisor implements a simple access control system
  - Subjects are partitions
  - Objects are simple hardware and hypervisor resources
    - memory, MSRs, memory mapped control registers, I/O ports
  - Basic support for DAC and MAC
- Complex and controversial functions not implemented in the hypervisor
  - Associating subjects (partitions) with meaningful security principals (e.g. users, code identities)
  - Concrete policy types (e.g. MLS, RBAC)
  - Will have to be implemented in a partition.

Control transfer

- Control transfer into the hypervisor
  - Hypervisor call, intercepts (traps), interrupts
- The hypervisor is non-preemptable
  - Interrupts are disabled while the hypervisor executes
  - All code paths must be short and predictable to avoid excessive interrupt latencies.
  - Also: No "worker VPs" waiting inside the hypervisor
  - BUT: Need to address concurrency because of multiprocessors
- Benefit: Simpler hypervisor (reduced state space)
- Difficulty: Constraints on interfaces and internal design

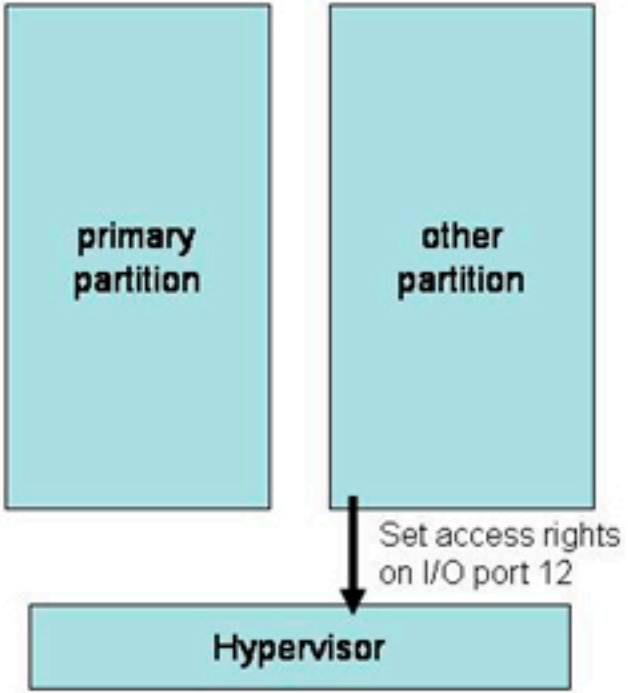
More on control transfer

- Further difficulties
  - Not all calls can be handled within the hypervisor
    - Intercepts (traps) routed to external handlers
    - In version 1: Calls may require the intervention of the primary partition (scheduler, memory manager)
  - The time spent in partitions is unpredictable
    - may take a long time
    - may never return

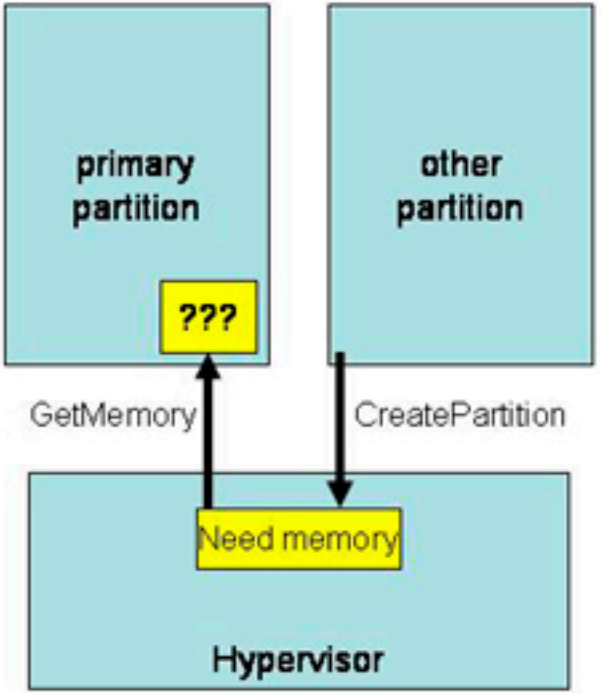
Easy case: call

-with short and predictable execution path

- that can be handled within the hypervisor



More difficult case: call that may involve other partitions



More on control transfer

- Only the hypervisor itself is non-preemptable.
  - Service partitions are preemptable
- The virtual processor that made the call
  - is blocked until the call completes
  - may take interrupts while the hypervisor is not executing